AMENDMENTS TO THE SPECIFICATION:

Please amend this application on page 1, line 1, by inserting the following new paragraph:

--This is a continuation of Application No. 09/615,316, filed July 13, 2000, which is incorporated herein by reference.--

Amend the paragraph bridging pages 5 and 6 to read as follows:

-- Control module 18 is also connected to a random access memory (RAM) 26, within which is stored a database 28, and to a program memory 30 which preferably constitutes a flash memory or other type of programmable non-volatile memory. Program memory 30 stores instructions for tasks executed by control module 18. Receiver 10 may also include an input device 32 such as a CD-ROM reader and a mass storage device 34 such as a hard disk. --

Amend the first paragraph at page 10 to read as follows:

-- Since there may be multiple instances of EIT0 tables, EIT source control structure 60 also includes a pointer to a second EIT source control structure 70 which in turn contains a pointer to an additional EIT source control structure 72. EIT source control structures 70 and 72 include pointers to Event dbEntries in the same manner as EIT source control structure 60. --

Amend the third paragraph at page 10 to read as follows:

-- Program memory 30 (Fig. 1) includes instructions for control module 18 to execute a method for maintaining database 28 for display of digital television broadcast

signals carried by a digital broadcast stream that includes content data and system control data. The system control data includes first information relating to a first broadcast standard and optionally includes second information relating to the second broadcast standard. --

Amend the first full paragraph at page 13 to read as follows:

-- When a dbEntry is created, the memchain control structure, such as 208, is first allocated and a unique memchain ID is assigned to the new chain/dbEntry. Then, a first portion of memory, for example, 210, is allocated and its address is stored as a pointer (for example, pointer 209) in a field of the control structure. Every time a new portion of memory is allocated, it is chained to the last allocated portion. Thus, portion 210 includes a pointer 215 in a known location with respect to the starting address of portion 210, independent of the actual amount of dbEntry data in portion 210. Similar pointers 216, 218, and ~~220~~ 219 are respectively contained in additional portions (memchain elements) 220, 222, and 224. Thus, in order to delete a dbEntry, all that is necessary is to read the memchain ID present in the dbEntry basic structure, access the corresponding memchain control structure, go through the pointer chain, and free the portions of memory one by one. The memory allocation and management function thus has no need to know the specific size and format of each individual memory portion. --

Amend the second full paragraph at page 14 to read as follows:

-- If the packet completes a PAT section, the method at stage 306 determines whether database 28 currently contains a transport dbEntry. If not, a transport dbEntry

is created in database 28 at stage 308, using the TSID of this PAT. At stage 310, the method then updates the MPEG control structure, including information such as version number, number of sections in the PAT, etc. A db progress update is then sent at stage 312 to all tasks of control module 18 which can generate queries to this portion of the database. The method then proceeds to a decode check at stage 360 to be described below. --

Amend the third full paragraph at page 14 to read as follows:

-- If it is determined in stage 306 that a transport dbEntry already exists in database 28, then a determination is made at stage 314 if the version number of the recently received PAT section is equal to the version number stored in the existing transport dbEntry. If so, the method proceeds to decode section 360. If not, the method, at stage 316, resets the MPEG control structure and disables all packet identifiers (PIDs) of previously received Program Mapping Table information. At stage 318, the MPEG control structure is updated with the data contained in the recently received PAT section. It is then determined at stage 320 if the transport stream identifier (TSID) contained in the recently received PAT section is equal to the TSID stored in the existing transport dbEntry. If so, the method proceeds to decode section 360. If not, the method removes the current transport dbEntry from database 28, creates a new transport dbEntry with the TSID from the recently received PAT section, and sends a database progress update, as indicated at stages 322, 324, and 326. --

Amend the first paragraph at page 15 to read as follows:

-- The method then executes a decode check at stage 360 (Fig. 5B) to determine if the recently received PAT section has been previously decoded. If so, the data has previously been stored in the database and the method returns to stage 300 to receive a new packet. If the PAT section has not been previously decoded, the method checks the next program number in the newly received PAT section and adds this program number to the channel list in the MPEG control structure, as shown in stage 362. The method then determines at stage 364 if this program number corresponds to a channel which is already present in the database. If not, the method, at stages 366 and 368, creates a new channel dbEntry and enables the PID of the Program Mapping Table for this particular channel, to permit future reception of PMT data. --

Amend the second paragraph at page 15 to read as follows:

-- After enabling the PMT PID or if the program number was already present in the channel database, the method determines at stage 370 if all program numbers of this PAT section have been processed. If not, processing of the next program number continues at stage 362. If this is the last program number in this PAT section, a determination is made at stage 372 if this is the last section of the PAT. If not, the method returns to stage 300 to receive the next packet. If this is the last PAT section, the method proceeds to stages 380-386 (Fig. 5C) to remove program dbEntries or programs no longer present in the transport stream. --

Amend the third paragraph at page 15 to read as follows:

-- Returning now to stage 302 (Fig. 5A), if the recently received packet is not a PAT packet, the method proceeds to stages 390, 392, and 394 (Fig. 5D) to store data from each program definition in a corresponding channel dbEntry if the packet is determined to be the last PMT packet in a section. If the recently received packet is not a PMT packet as determined in stage 390, the method determines at stage 396 if the recently received packet is a PSIP packet. If not, the packet is determined to be a packet other than a system data control packet and it is processed by other methods not relevant to the present invention. --

Amend the first full paragraph at page 16 to read as follows:

-- At stage 398, if the recently received packet is not an MGT packet and is a VCT packet, the VCT data is processed and stored in database 28, as indicated by stages 412-442 (Figs. 5E - 5F). In particular, the method performs a coherency check at stage 438 to determine the recently received VCT data is consistent with previously stored MPEG data. Specifically, the coherency check verifies that the number and type of elementary streams for each program are the same in the PSIP tables and the MPEG tables. --

Amend the second full paragraph at page 16 to read as follows:

-- If the data is determined to be consistent, the channel information in the recently received data is stored in database 28 information at stage ~~440~~ 446, merged with existing MPEG channel information. On the other hand, if the recently received

VCT data is not consistent with previously stored MPEG data, the method resets the decoding status of this VCT section in the PSIP control structure 44 at stage 442 such that the VCT data is ignored. --

Amend the third full paragraph at page 16 to read as follows:

-- If the recently received packet is not a VCT packet, as determined in stage 412 (Fig. 5E), the method determines if the packet is an EIT0 packet or an ETT0 packet, at stages 460 and 480 (Fig. 5G), respectively. EIT0 packets are processed by stages 462-479 and ETT0 packets are processed by stages 482-492 (Fig. 5H). If the packet is not an ETT0 packet, as determined at stage 480, then the packet is processed as either an RRT packet or an STT packet, in a manner not relevant to the present invention. --

AMENDMENTS TO THE DRAWINGS:

Attached are two (2) sheets of drawings:

1) One (1) Replacement Sheet for Fig. 4.

2) One (1) Informal Mark-Up Sheet for Fig. 4.